Problem 5-1 (BPP and NP) 15 points

Show that if $\text{NP} \subseteq \text{BPP}$, then $\text{NP} = \text{RP}$.

Problem 5-2 ($x^x = x$ for $x \in \{\text{BPP}, \text{RP}\}$) 10 points

Let $\text{BPP}^{\text{BPP}} = \bigcup_{B \in \text{BPP}} \text{BPP}^B$. Show that $\text{BPP}^{\text{BPP}} = \text{BPP}$.

Is $\text{RP}^{\text{RP}} = \text{RP}$? Why or why not?

Problem 5-3 (ZPP = RP $\cap$ coRP) 20 points

Show that $\text{ZPP} = \text{RP} \cap \text{coRP}$. Remember to argue both directions.

(Hint: Remember Markov’s inequality: if random variable $X > 0$, then $\Pr(X > t\mathbb{E}[X]) \leq 1/t$.)

Problem 5-4 (Logarithmic Advice) 15 points

Recall the class $\text{P/\hspace{-0.5em}poly}$ of languages accepted by polynomial time Turing machines having polynomially long (in the input size) advice string. We saw that $\text{BPP} \subseteq \text{P/\hspace{-0.5em}poly}$, and mentioned that it is very unlikely that $\text{NP} \subseteq \text{P/\hspace{-0.5em}poly}$. Define a more restrictive class $\text{P/log}$ which is the same as $\text{P/poly}$ except the advice is only logarithmic in the input size (i.e., there could be only polynomially many advice strings for each input length). Show that $\text{NP} \subseteq \text{P/log}$ implies that $\text{P} = \text{NP}$.

Problem 5-5 (Testing Equality) 15 points

Let $a = a_1 \ldots a_n$, $b = b_1 \ldots b_n \in \{0, 1\}^n$ be two $n$-bit numbers held by Alice and Bob respectively. Alice and Bob try to determine if $a = b$ using the smallest amount of communication. We mentioned that the deterministic protocol requires $n$ bits of communication. Here you will compare two probabilistic protocols.

(a) Alice and Bob interpret $a_i$’s and $b_i$’s as coefficients of degree $n$ polynomials $A(x) = \sum_i a_i x^{i-1}$ and $B(x) = \sum_i b_i x^{i-1}$ over a field $\mathbb{Z}_p$, where $p > n^2$ is a prime. Then Alice picks (after choosing $p$) a random $x \in \mathbb{Z}_p$ and sends $p, x, A(x)$ to Bob, and Bob agrees if $A(x) = B(x)$.

This requires $6 \log n$ bits of communication. If $a = b$, then Bob accepts. Assume $a \neq b$. What is the maximum probability Bob accepts (this is called the error probability)?

(b) Alice picks a random prime $p$ of length $s$ bits (to be determined), and sends $p$ together with $a \mod p$. Bob agrees if $a \mod p = b \mod p$. What is the value of $s$ so that the error probability
of this test is less than in the previous protocol? How does your communication complexity compare with the previous protocol?\(^1\)

(Hint: Use the prime number theorem saying that there are approximately \(k/\log k\) primes less than \(k\).)

**Problem 5-6 (Game Tree Evaluation) \hspace{1cm} 30 points**

This problem illustrates the differences between deterministic, non-deterministic and randomized computation. A problem of game-tree evaluation is the following. One is given a complete binary tree \(T\) of depth \(2^k\). Each internal node contains an \(\land\) (AND) or an \(\lor\) (OR) gate. The gates of \(T\) “alternate”: the two children of the \(\lor\) gate are \(\land\) gates (except if the children are the leaves), and vice versa (i.e., the only freedom is to choose the root, other internal gates are determined from there on). Each of the \(n = 2^{2^k}\) leaves of \(T\) contains a boolean value (1 = true or 0 = false). The value of the tree \(T\) is the value of of the root when the tree is evaluated in a natural fashion. In one time step your algorithm (specified later) is allowed to read the value of the leaf that it specifies, all other computation is free.

Now, a deterministic algorithm reads \(l\) leaves and must be correct for all the leaf assignments (of course, all its decisions are deterministic). Its time is the worst-case number \(l\) of leaves that it reads.

A randomized algorithm is allowed to flip random coins when determining which \(l\) leaves to read, but it must always give a correct answer (i.e., be “ZPP”-like). Its complexity is the worst-case (over the input assignments) expected number of leaves \(l\) that it reads.

Finally, a non-deterministic algorithm can non-deterministically guess which \(l\) leaves to read. Each non-deterministic branch has to either reject, or output the correct tree value + at least one branch should not reject. The complexity of the algorithm is the smallest number of leaves read by the non-rejecting (i.e., correct) branch (if you wish, you can reject all the branches reading more than this smallest number of leaves), taken over the worst-case leaf assignment.

(a) Show (by induction on \(k\)) that any deterministic algorithm must read all \(n = 4^k\) leaves.

(b) Consider the non-deterministic algorithm that operates as follows. When evaluating an \(\lor\) gate, non-deterministically choose one of the two children and recursively evaluate it. If the value is 1, ignore the second child and output 1. If the value is 0, it recursively evaluates the second child and returns its value. Similarly for the \(\land\) gate with 0 replaced by 1. Let \(W(k)\) be the complexity of this algorithm when the root is an \(\lor\) gate. Similarly, let \(L(k)\) be the complexity when the root is an \(\land\) gate.

Write recurrences for \(W(k)\) and \(L(k)\) and show that \(W(k) = L(k) = 2^k = \sqrt{n}\).

(c) Consider the randomized algorithm that operates in the same manner as the non-deterministic algorithm except it chooses a child at random rather than non-deterministically. Again, write the recurrence for \(W(k)\) and \(L(k)\) and show that \(W(k) = L(k) = 3^k \approx n^{0.793}\).

---

\(^1\)Food for thought. The first protocol fixes the field (as long as it is large enough), and picks a random point \(x\) in the field. In the second protocol, computing \((a \mod p)\) and \((b \mod p)\) can be viewed as evaluating \((A(2) \mod p)\) and \((B(2) \mod p)\) respectively. Incidentally, one can see that the point \(x = 2\) is arbitrary, any point \(x \neq 1\) works as long as it is “small enough” (why?). Thus, the field is now chosen at random, while the point is fixed (as long as it is small).
Extra 1 (?Zero-Knowledge? Interactive Protocols) 30 points

Recall that \( \mathbb{Z}_p = \{0, \ldots, p-1\} \) — the set of numbers modulo \( p \) — is a field when \( p \) is a prime. Moreover, its multiplicative group \( \mathbb{Z}_p^* = \{1, \ldots, p-1\} \) is cyclic: it has a generator \( g \), that is \( \{g^1 \mod p, \ldots, g^{p-1} \mod p\} = \mathbb{Z}_p^* \). Recall also that the order of an element \( x \in \mathbb{Z}_p^* \) is the smallest integer \( \alpha > 0 \) such that \( x^\alpha \mod p = 1 \). It is well known that the order of every \( x \) must divide \((p-1)\), which is the size of \( \mathbb{Z}_p^* \). Thus, a generator has order \((p-1)\), while every other element has order at most \((p-1)/2\). Also, if \( g \) is a generator and \( g^k = x \), we say that \( k \) is the index of \( x \) (w.r.t. \( g \)) and write \( k = \text{ind}(x) \) (\( g \) is implicit here). Note, \( g \) being a generator means \( \text{ind}(x) \) is well defined for every \( x \in \mathbb{Z}_p^* \). Recall also that computing \((g^k \mod n)\) can be done in polynomial time, while computing the index of \( x \) is believed to be very hard.

Let \( L = \{(p,g) \mid p - \text{prime}, g - \text{generator for } \mathbb{Z}_p^*\} \). In this problem we will attempt to design zero-knowledge interactive protocols for \( L \). Therefore, for each of the protocols below, do the following in this order: (1) formally prove that the resulting protocol is an interactive protocol for \( L \) (that is, is it complete and sound)? Be brief, since correct proofs are very short; (2) answer if the resulting protocol is zero-knowledge against the honest verifier? This part can be informal, if you wish, but try to give reasons behind your answer; (3) answer if the resulting protocol is zero-knowledge against even dishonest verifier. This part certainly does not have to be formal, but try to come up with a good reason for your answer.

In the protocols below \( P \) stands for the prover, \( V \) – for the verifier, the common input is a pair \((p,g)\), and all the computation is modulo \( p \).

Protocol 1:

- \( P \) sends to \( V \) the factorization of \( p - 1 = p_1^{\alpha_1} \ldots p_k^{\alpha_k} \), where \( p_i \)'s are distinct primes and \( \alpha_i > 0 \).
- \( V \) checks if \( p, p_1, \ldots, p_k \) are primes, \( p - 1 = p_1^{\alpha_1} \ldots p_k^{\alpha_k} \) and \( g^{(p-1)/p_i} \neq 1 \) for \( i = 1 \ldots k \).

Protocol 2:

- \( V \) checks if \( p \) is a prime. If not, he rejects. Else, \( V \) picks a random \( y \) in \( \mathbb{Z}_p^* \) and sends \( y \) to \( P \).
- \( P \) stops if \( y \notin \mathbb{Z}_p^* \). \( P \) finds \( x = \text{ind}(y) \) and sends \( x \) to \( V \).
- \( V \) verifies that \( g^x = y \mod p \).

Protocol 3:

- \( V \) checks if \( p \) is a prime. If not, he rejects. Else, \( V \) picks a random \( x \) in \( \mathbb{Z}_p^* \) and sends \( x \) to \( P \).
- \( P \) stops if \( x \notin \mathbb{Z}_p^* \). Else \( P \) picks a random \( y \) in \( \mathbb{Z}_p^* \), and sends \( y \) to \( V \).
- \( V \) flips a coin \( c \) and sends \( c \) to \( P \).
- \( P \) stops if \( c \) is not a bit. If \( c = 0 \), \( P \) sets \( z = \text{ind}(y) \), else \( z = \text{ind}(xy) \). \( P \) sends \( z \) to \( V \).
- \( V \) verifies the result by checking that \( g^z = y \) when \( c = 0 \), or that \( g^z = xy \) when \( c = 1 \).

Protocol 4: (extra-credit, optional)
• $V$ checks if $p$ is a prime. If not, he rejects.
• $P$ picks a random $a \in \mathbb{Z}_p^*$, and sends $x = g^a$ to $V$.
• $V$ picks a random $b$ in $\mathbb{Z}_{p-1}$, and sends $y = x^b$ to $P$.
• $P$ computes $z = \text{ind}(y) \cdot a^{-1} \mod (p - 1)$, and sends $z$ to $V$.
• $V$ verifies that $b = z$. 